Linked List Utilities

We have been talking about ordered linked lists and unordered linked lists. In this assignment, you will be blurring the line between those by writing helper methods for a generic LinkedList class that allows you to manipulate both kinds of linked lists.

- You will want to reuse your definition of class Node.
- You will find it helpful to cannibalize from the definitions of the methods we defined for the classes unorderedList and orderedList.
- **For this assignment, you should assume that all information stored in the "data" field of each node points to a string, NOT an integer.**

The methods you must define for the class LinkedList include:

```
def __str__ (self):
   # Return a string representation of data suitable for printing.
   #    Long lists (more than 10 elements long) should be neatly
   #    printed with 10 elements to a line, two spaces between
   #    elements

def addFirst (self, item):
   # Add an item to the beginning of the list

def addLast (self, item):
   # Add an item to the end of a list

def addInOrder (self, item):
   # Insert an item into the proper place of an ordered list.
   # This assumes that the original list is already properly
   #    ordered.

def getLength (self):
   # Return the number of items in the list

def findUnordered (self, item):
   # Search in an unordered list
   #    Return True if the item is in the list, False
   #    otherwise.

def findOrdered (self, item):
   # Search in an ordered list
   #    Return True if the item is in the list, False
   #    otherwise.
   # This method MUST take advantage of the fact that the
   #    list is ordered to return quicker if the item is not
   #    in the list.

def delete (self, item):
   # Delete an item from an unordered list
   #    if found, return True; otherwise, return False
```

```
  def copyList (self):
     # Return a new linked list that's a copy of the original,
     #    made up of copies of the original elements

  def reverseList (self):
     # Return a new linked list that contains the elements of the
     #    original list in the reverse order.

  def orderList (self):
     # Return a new linked list that contains the elements of the
     #    original list arranged in ascending (alphabetical) order.
     #    Do NOT use a sort function:  do this by iteratively
     #    traversing the first list and then inserting copies of
     #    each item into the correct place in the new list.

  def isOrdered (self):
     # Return True if a list is ordered in ascending (alphabetical)
     #    order, or False otherwise

  def isEmpty (self):
     # Return True if a list is empty, or False otherwise

  def mergeList (self, b):
     # Return an ordered list whose elements consist of the
     #    elements of two ordered lists combined.

  def isEqual (self, b):
     # Test if two lists are equal, item by item, and return True.

  def removeDuplicates (self):
     # Remove all duplicates from a list, returning a new list.
     #    Do not change the order of the remaining elements.
```

Main Program:

You do not need to write a main program for this assignment. Instead, download the main program from here:

```
# Copy and paste the following after your class definitions for
# Nodes and LinkedLists.  Do NOT change any of the code in main()!

def main():

   print
("\n\n***************************************************************")
   print ("Test of addFirst:  should see 'node34...node0'")
   print ("***************************************************************")
   myList1 = LinkedList()
   for i in range(35):
      myList1.addFirst("node"+str(i))

   print (myList1)

   print
("\n\n***************************************************************")
```

```
   print ("Test of addLast:  should see 'node0...node34'")
   print ("****************************************************************")
   myList2 = LinkedList()
   for i in range(35):
      myList2.addLast("node"+str(i))

   print (myList2)

   print
("\n\n****************************************************************")
   print ("Test of addInOrder:  should see 'alpha delta epsilon gamma
omega'")
   print ("****************************************************************")
   greekList = LinkedList()
   greekList.addInOrder("gamma")
   greekList.addInOrder("delta")
   greekList.addInOrder("alpha")
   greekList.addInOrder("epsilon")
   greekList.addInOrder("omega")
   print (greekList)

   print
("\n\n****************************************************************")
   print ("Test of getLength:  should see 35, 5, 0")
   print ("****************************************************************")
   emptyList = LinkedList()
   print ("   Length of myList1:  ", myList1.getLength())
   print ("   Length of greekList:  ", greekList.getLength())
   print ("   Length of emptyList:  ", emptyList.getLength())

   print
("\n\n****************************************************************")
   print ("Test of findUnordered:  should see True, False")
   print ("****************************************************************")
   print ("   Searching for 'node25' in myList2:
",myList2.findUnordered("node25"))
   print ("   Searching for 'node35' in myList2:
",myList2.findUnordered("node35"))

   print
("\n\n****************************************************************")
   print ("Test of findOrdered:  should see True, False")
   print ("****************************************************************")
   print ("   Searching for 'epsilon' in greekList:
",greekList.findOrdered("epsilon"))
   print ("   Searching for 'omicron' in greekList:
",greekList.findOrdered("omicron"))

   print
("\n\n****************************************************************")
   print ("Test of delete:  should see 'node25 found', 'node34 found',")
   print ("   'node0 found', 'node40 not found'")
   print ("****************************************************************")
   print ("   Deleting 'node25' (random node) from myList1: ")
   if myList1.delete("node25"):
      print ("      node25 found")
   else:
      print ("      node25 not found")
   print ("   myList1:  ")
   print (myList1)

   print ("   Deleting 'node34' (first node) from myList1: ")
   if myList1.delete("node34"):
      print ("      node34 found")
   else:
      print ("      node34 not found")
   print ("   myList1:  ")
   print (myList1)

   print ("   Deleting 'node0'  (last node) from myList1: ")
   if myList1.delete("node0"):
      print ("      node0 found")
   else:
      print ("      node0 not found")
   print ("   myList1:  ")
   print (myList1)

   print ("   Deleting 'node40' (node not in list) from myList1: ")
   if myList1.delete("node40"):
      print ("      node40 found")
   else:
      print ("   node40 not found")
   print ("   myList1:  ")
   print (myList1)

   print
("\n\n**********************************************************")
   print ("Test of copyList:")
   print ("**********************************************************")
   greekList2 = greekList.copyList()
   print ("   These should look the same:")
   print ("      greekList before delete:")
   print (greekList)
   print ("      greekList2 before delete:")
   print (greekList2)
   greekList2.delete("alpha")
   print ("   This should only change greekList2:")
   print ("      greekList after deleting 'alpha' from second list:")
   print (greekList)
   print ("      greekList2 after deleting 'alpha' from second list:")
   print (greekList2)
   greekList.delete("omega")
   print ("   This should only change greekList1:")
   print ("      greekList after deleting 'omega' from first list:")
   print (greekList)
   print ("      greekList2 after deleting 'omega' from first list:")
   print (greekList2)

   print
("\n\n**********************************************************")
   print ("Test of reverseList:  the second one should be the reverse")
   print ("**********************************************************")
   print ("   Original list:")
   print (myList1)
   print ("   Reversed list:")
   myList1Rev = myList1.reverseList()
   print (myList1Rev)

   print
("\n\n***************************************************************")
   print ("Test of orderList:  the second list should be the first one
sorted")
   print ("***************************************************************")
   planets = LinkedList()
   planets.addFirst("Mercury")
   planets.addFirst("Venus")
   planets.addFirst("Earth")
   planets.addFirst("Mars")
   planets.addFirst("Jupiter")
   planets.addFirst("Saturn")
   planets.addFirst("Uranus")
   planets.addFirst("Neptune")
   planets.addFirst("Pluto?")

   print ("   Original list:")
   print (planets)
   print ("   Ordered list:")
   orderedPlanets = planets.orderList()
   print (orderedPlanets)

   print
("\n\n***************************************************************")
   print ("Test of isOrdered:  should see False, True")
   print ("***************************************************************")
   print ("   Original list:")
   print (planets)
   print ("   Ordered? ", planets.isOrdered())
   orderedPlanets = planets.orderList()
   print ("   After ordering:")
   print (orderedPlanets)
   print ("   ordered? ", orderedPlanets.isOrdered())

   print
("\n\n***************************************************************")
   print ("Test of isEmpty:  should see True, False")
   print ("***************************************************************")
   newList = LinkedList()
   print ("New list (currently empty):", newList.isEmpty())
   newList.addFirst("hello")
   print ("After adding one element:",newList.isEmpty())

   print
("\n\n***************************************************************")
   print ("Test of mergeList")
   print ("***************************************************************")
   list1 = LinkedList()
   list1.addLast("aardvark")
   list1.addLast("cat")
   list1.addLast("elephant")
   list1.addLast("fox")
   list1.addLast("lynx")
   print ("   first list:")
   print (list1)
   list2 = LinkedList()
   list2.addLast("bacon")
   list2.addLast("dog")
   list2.addLast("giraffe")
   list2.addLast("hippo")
   list2.addLast("wolf")
   print ("   second list:")
   print (list2)
   print ("   merged list:")
   list3 = list1.mergeList(list2)
   print (list3)

   print
("\n\n**************************************************************")
   print ("Test of isEqual:  should see True, False, True")
   print ("**************************************************************")
   print ("   First list:")
   print (planets)
   planets2 = planets.copyList()
   print ("   Second list:")
   print (planets2)
   print ("      Equal:  ",planets.isEqual(planets2))
   print (planets)
   planets2.delete("Mercury")
   print ("   Second list:")
   print (planets2)
   print ("      Equal:  ",planets.isEqual(planets2))
   print ("   Compare two empty lists:")
   emptyList1 = LinkedList()
   emptyList2 = LinkedList()
   print ("      Equal:  ",emptyList1.isEqual(emptyList2))

   print
("\n\n**************************************************************")
   print ("Test of removeDuplicates:  original list has 14 elements, new list
has 10")
   print ("**************************************************************")
   dupList = LinkedList()
   print ("   removeDuplicates from an empty list shouldn't fail")
   newList = dupList.removeDuplicates()
   print ("   printing what should still be an empty list:")
   print (newList)
   dupList.addLast("giraffe")
   dupList.addLast("wolf")
   dupList.addLast("cat")
   dupList.addLast("elephant")
   dupList.addLast("bacon")
   dupList.addLast("fox")
   dupList.addLast("elephant")
   dupList.addLast("wolf")
   dupList.addLast("lynx")
   dupList.addLast("elephant")
   dupList.addLast("dog")
   dupList.addLast("hippo")
   dupList.addLast("aardvark")
   dupList.addLast("bacon")
   print ("   original list:")
   print (dupList)
   print ("   without duplicates:")
   newList = dupList.removeDuplicates()
   print (newList)

main()
```

Attach the contents of this file to the class definitions and functions that you have written, and run the file. Correct any errors that occur in YOUR code. **Do NOT change the main program! It is written exactly the way we want it.**

Input:

There is no data file for this program.

Output:

The output of your program will be generated by the main program, so you do not need to include any print statements in your code. Be sure to carefully review the output you get to ensure your code is behaving correctly.